Russ Perry, Technical Sales Specialist, Mid-Atlantic

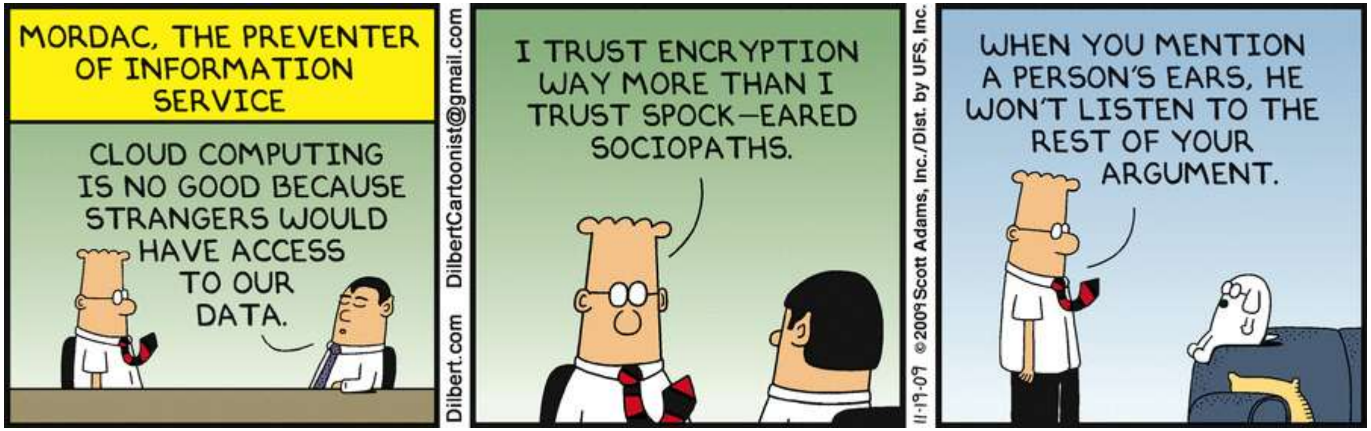
August 19, 2015

DB2 Native Encryption

DB2 Native Encryption

3 Main Points

- Helps companies address compliance and security requirements
- Native, implemented as part of DB2
- Simple to implement and manage


MORDAC, THE PREVENTER OF INFORMATION SERVICE
CLOUD COMPUTING IS NO GOOD BECAUSE STRANGERS WOULD HAVE ACCESS TO OUR DATA.
Dilbert.com DilbertCartoonist@gmail.com
I TRUST ENCRYPTION WAY MORE THAN I TRUST SPOCK–EARED SOCIOPATHS.
11-19-09 ©2009 Scott Adams, Inc./Dist. by UFS, Inc.
WHEN YOU MENTION A PERSON'S EARS, HE WON'T LISTEN TO THE REST OF YOUR ARGUMENT.

Agenda

Why Should We Encrypt Our Databases?

IBM DB2 Native Encryption Overview

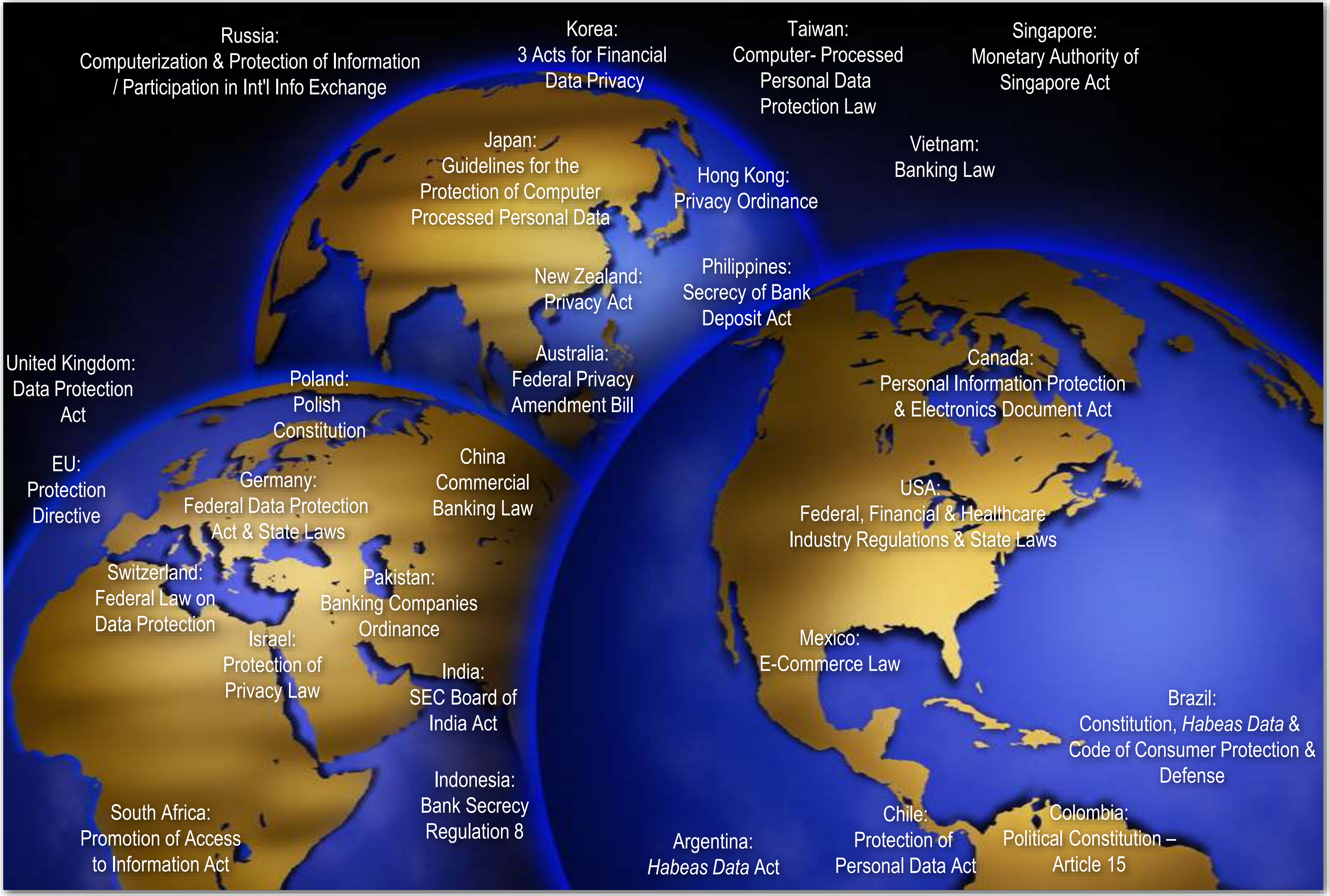
Encryption key management

Encrypting DB2 databases

Backup and Restore

Utilities, diagnostics, and other considerations

Encryption is more than just good business - Often times it is the law



Russia:
Computerization & Protection of Information / Participation in Int'l Info Exchange

Korea:
3 Acts for Financial Data Privacy

Taiwan:
Computer- Processed Personal Data Protection Law

Singapore:
Monetary Authority of Singapore Act

Japan:
Guidelines for the Protection of Computer Processed Personal Data

Hong Kong:
Privacy Ordinance

Vietnam:
Banking Law

New Zealand:
Privacy Act

Philippines:
Secrecy of Bank Deposit Act

Australia:
Federal Privacy Amendment Bill

United Kingdom:
Data Protection Act

Poland:
Polish Constitution

Canada:
Personal Information Protection & Electronics Document Act

EU:
Protection Directive

Germany:
Federal Data Protection Act & State Laws

China
Commercial Banking Law

USA:
Federal, Financial & Healthcare Industry Regulations & State Laws

Switzerland:
Federal Law on Data Protection

Pakistan:
Banking Companies Ordinance

Israel:
Protection of Privacy Law

India:
SEC Board of India Act

Mexico:
E-Commerce Law

Brazil:
Constitution, *Habeas Data* & Code of Consumer Protection & Defense

Indonesia:
Bank Secrecy Regulation 8

South Africa:
Promotion of Access to Information Act

Argentina:
Habeas Data Act

Chile:
Protection of Personal Data Act

Colombia:
Political Constitution – Article 15

Why Use Data Encryption?
General requirements

1. Helps Companies meet compliance requirements
   - ➢ Industry standards such as PCI DSS
   - ➢ Regulations such as SarbOx, HIPAA
   - ➢ Corporate standards
2. Protect against threats to online data
   - ➢ Users accessing database data outside the scope of the DBMS
3. Protect against threats to offline data
   - ➢ Theft or loss of physical media

DB2 Note:

DB2 native encryption offers many advantages:
- Reduces cost of security and compliance
- Eliminates the need for third-party add-on tools
- Is easily used by DB2 bundlers such as ISVs
- Runs wherever DB2 LUW runs

Agenda

Why Should We Encrypt Our Databases?

IBM DB2 Native Encryption Overview

Encryption key management

Encrypting DB2 databases

Backup and Restore

Utilities, diagnostics, and other considerations

IBM DB2 Native Encryption
Key Points

- DB2 Native Encryption is part of the DB2 database server core capabilities
  - ➢ As of version 10.5 - Fix Pack 5
  - ➢ Runs on all 64-bit platforms: AIX, HP-UX, Linux, pLinux, zLinux, Solaris, Windows
  - ➢ Exploits available HW acceleration (AES encryption only)
- Provides a cost-effective encryption compliance method
  - ➢ As an advanced edition feature or available separately
  - ➢ Requires no hardware or software changes
  - ➢ Provides a secure key management solution
- Protects against physical theft of disk devices as well as backup images
  - ➢ Using Public Key Cryptography Standard #12 (PKCS#12)
- Requires no schema or application changes
  - ➢ Is transparent to the end users and applications (requires no changes)
- Compliant, e.g.
  - ➢ NIST SP 800-131 compliant cryptographic algorithms
  - ➢ Uses FIPS 140-2 certified encryption

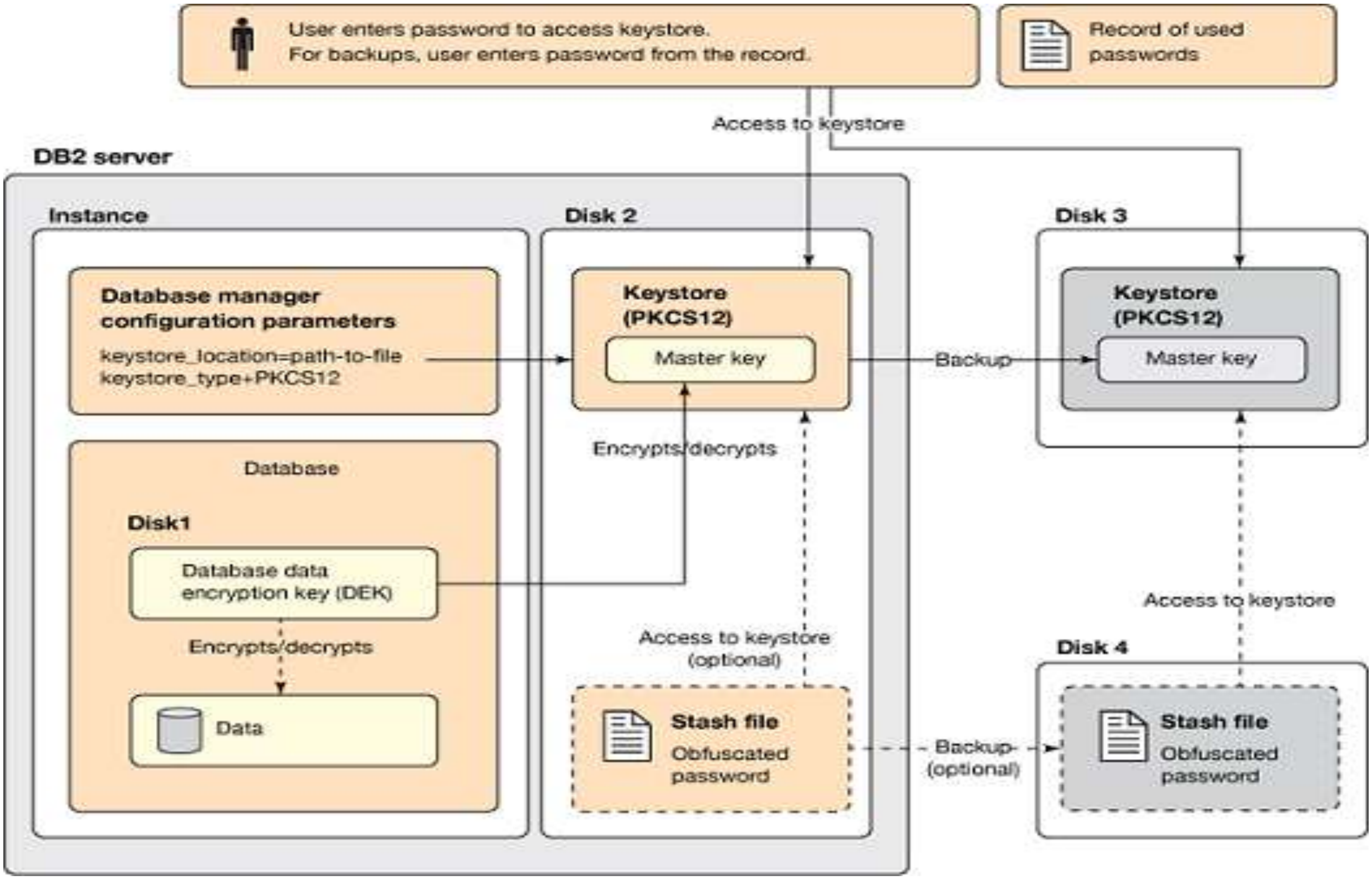
IBM DB2 Native Encryption Scope
How it works

- The engine encrypts the data before it calls the file system to write to disk
  - Current and future data is protected
  - A decryption occurs during reads from the file system
- Data is protected in:
  - Table space containers (all types and all data, including LOB, XML, etc.)
  - Transaction logs
  - LOAD copy and LOAD staging tables
  - Dump files
  - Backup images
- The data encryption is done using a data encryption key
  - This DB key is stored and managed in the database itself
- A master encryption key protects the data encryption key
  - This master key is stored outside the database in a keystore

Encryption Key Wrapping

- The process of encrypting one key with another key
- The key encrypting key is typically referred to as a **Master Key**(MK)
- The MK is typically stored separately from the data system
- The top drivers for this 2- tier encryption approach are:
  - **Security**: Compromise of the data system does not mean compromise of the data as the MK is stored outside the data system
  - **Performance**: Complying with key rotation requirements **does not mean re-encrypting the actual data**; only the **Data Encryption Key**(DEK) is re-encrypted with a new MK
- DB2 Implements the industry standard 2-tier model
- Actual data is encrypted with a DEK
- DEK is encrypted with a MK

IBM DB2 Native Encryption Overview

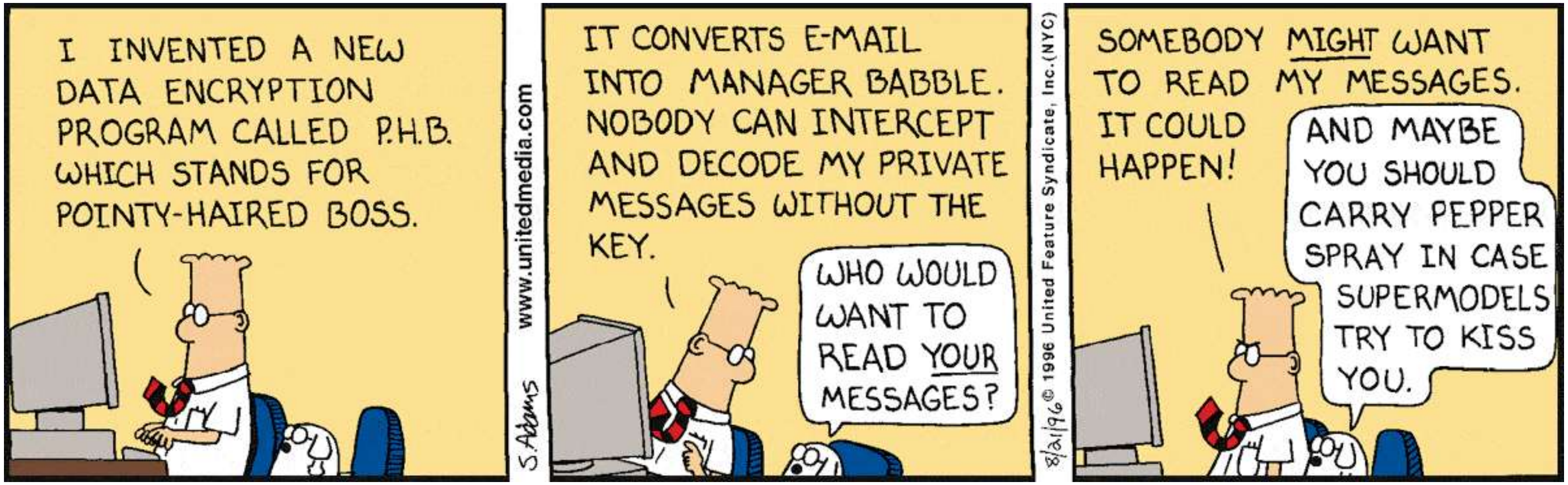


Source: www.ibm.com/developerworks/data/library/techarticle/dm-1504-master-encrypted-keys

IBM DB2 Encryption Offering Licensing

- Included in these Editions
  - DB2 Advanced Enterprise Server
  - DB2 Advanced Workgroup Server
  - Express-C

- License available for these editions:
  - DB2 Enterprise Server
  - DB2 Workgroup Server
  - DB2 Express Server


I INVENTED A NEW DATA ENCRYPTION PROGRAM CALLED P.H.B. WHICH STANDS FOR POINTY-HAIRED BOSS.
www.unitedmedia.com
S. Adams
IT CONVERTS E-MAIL INTO MANAGER BABBLE. NOBODY CAN INTERCEPT AND DECODE MY PRIVATE MESSAGES WITHOUT THE KEY.
WHO WOULD WANT TO READ YOUR MESSAGES?
8/21/96 © 1996 United Feature Syndicate, Inc.(NYC)
SOMEBODY MIGHT WANT TO READ MY MESSAGES. IT COULD HAPPEN!
AND MAYBE YOU SHOULD CARRY PEPPER SPRAY IN CASE SUPERMODELS TRY TO KISS YOU.

Agenda

Why Should We Encrypt Our Databases?

IBM DB2 Native Encryption Overview

Encryption key management

Encrypting DB2 databases

Backup and Restore

Utilities, diagnostics, and other considerations

IBM Global Security Kit

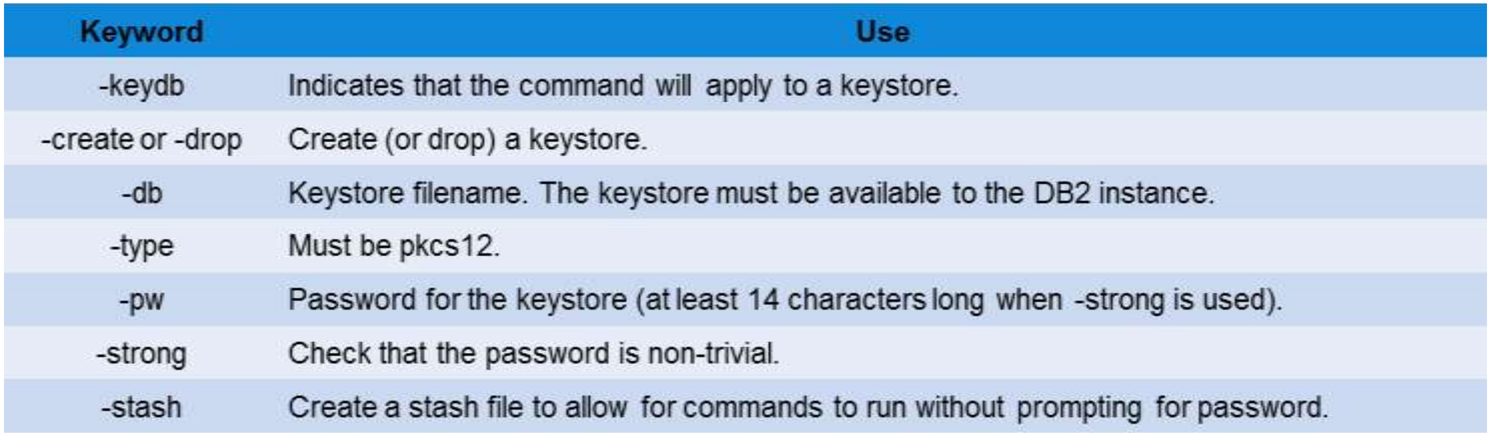
- DB2 Native Encryption uses the IBM Global Security Kit (GSKit) for key management and encryption
  - Installed with DB2 in the sqllib/gskit directory
  - GSKit libraries are used to encrypt/decrypt data, create store and manage MKs
  - FIPS 140-2 certified
  - gsk8capicmd_64 is the command line tool used to manage the keystore

- Public Key Cryptography Standard (PKCS) #12:
  - A password-protected keystore with a format for storing encryption keys
  - Local keystore file
  - Stores MKs
  - Can use the same keystore for SSL certificates

Keystore creation

- The first step to implementing DB2 Native Encryption is to create a keystore
  - Example:

    -
      ```
      gsk8capicmd_64 -keydb -create -db ~/db2/db2keys.p12
                     -type pkcs12
                     -pw "Str0ngPassw0rd"
                     -strong -stash
      ```



Keyword	Use
-keydb	Indicates that the command will apply to a keystore.
-create or -drop	Create (or drop) a keystore.
-db	Keystore filename. The keystore must be available to the DB2 instance.
-type	Must be pkcs12.
-pw	Password for the keystore (at least 14 characters long when -strong is used).
-strong	Check that the password is non-trivial.
-stash	Create a stash file to allow for commands to run without prompting for password.

Stash File Considerations

- When the -stash option is specified during the create action, an obfuscated version of the keystore password is stashed in a file:
  - <key database name>.sth
- A stash file is used as an automatic way of providing a password
  - If a keystore password was not provided during db2start, the password will be retrieved from the stash file
- The stash file can only be read by the instance owner
  - Not stashing the password enhances security if the instance owner account becomes compromised
  - This additional security must be weighed against any requirements that the DB2 instance can start without human intervention
  - If the password is not stashed, you cannot access an encrypted database until you provide the keystore password.

Starting DB2 without a Stash File

- DB2 will start normally (no error condition returned) if a stash file is not present in the system
- Database activation, or applications connecting to encrypted databases will encounter an error condition:

```
       SQL1728N  The command or operation failed because the keystore
could not be accessed. Reason code "3".
```

- The db2start command must be re-executed with the open keystore option to enable access to encrypted databases

```
       db2start open keystore USING KeySt0rePassw0rd
```

- To avoid placing the password on the command line:

`db2start open keystore` ← will prompt for password

- For scripts of other executables, supply the password through either a temporary file or open file descriptor

```
       db2start open keystore PASSARG [FILENAME:<value> | FD:<value>]
```

Creating Master Keys

- DB2 may generate MKs for you automatically during:
  - Database Creation
  - Key rotation
  - Restoring into a new database
  - Default is AES 256-bit
- This key is used to encrypt the DEK, not the actual database
- You may want to create a MK with a specific label for a number of reasons:
  - You want to keep track of the Master Key Labels and their corresponding keys for offsite recovery without having the entire keystore available on the backup site
  - You have an HADR pair that must have synchronized keys
  - You are encrypting a backup for an unencrypted database

Creating Master Keys

- A secret key needs to be generated by the user before adding a master key to the keystore
  - The secret key is used to encrypt the DEK
  - The strength of the secret key has no relationship to the actual encryption that takes place within the database
  - Recommendation is to use the highest level of AES encryption (256) for the MK

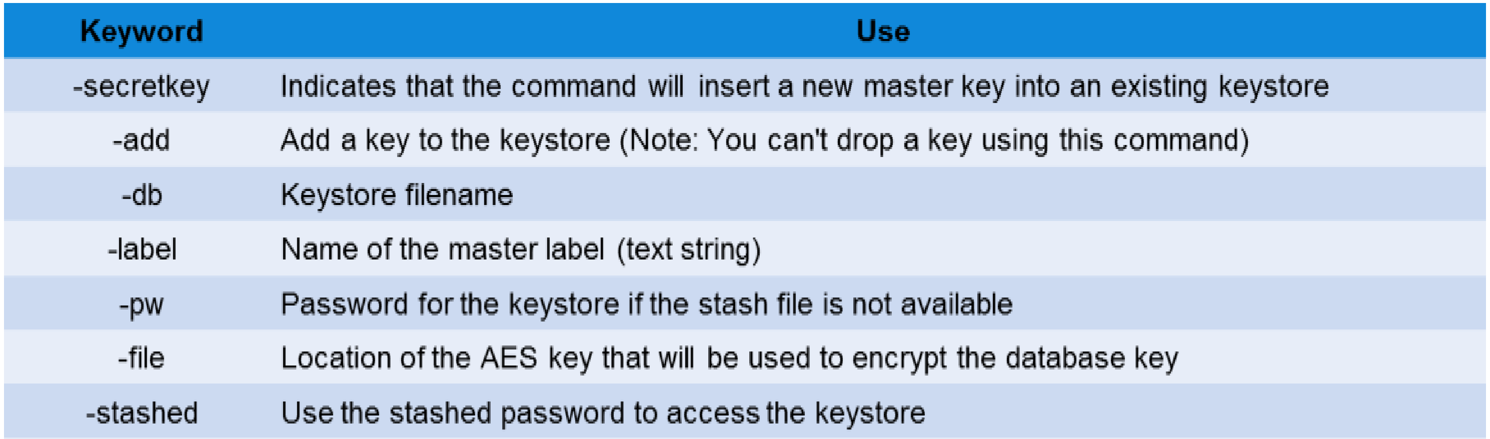
- Generating a random key
  - A key needs to be 16, 24, or 32 bytes wide
    - Corresponds to 128, 192, or 256-bit AES keys
  - On Linux, UNIX, and AIX use the following command to generate a 32-byte random string (which will become our MK)

```
head -c 32 /dev/random >~/db2/mysecretkey
```

Creating Master Keys

- A Master Key Label is used to refer to a Master Key
  - Example:

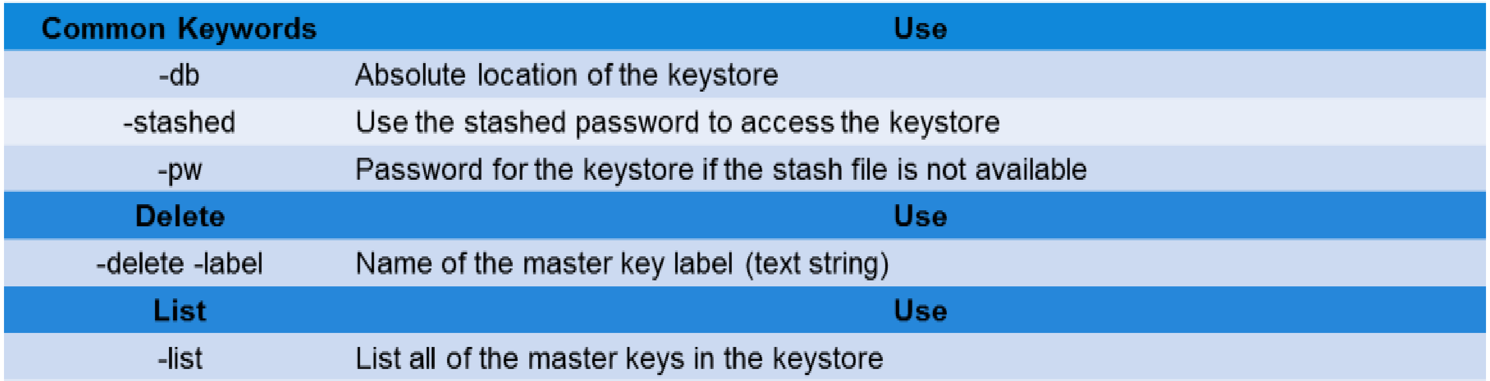
```
• gsk8capicmd_64 -secretkey -add -db ~/db2/db2keys.p12
                 -label secret.key
                 -stashed
                 -file ~/db2/mysecretkey
```



Keyword	Use
-secretkey	Indicates that the command will insert a new master key into an existing keystore
-add	Add a key to the keystore (Note: You can't drop a key using this command)
-db	Keystore filename
-label	Name of the master label (text string)
-pw	Password for the keystore if the stash file is not available
-file	Location of the AES key that will be used to encrypt the database key
-stashed	Use the stashed password to access the keystore

Listing Master Keys

- You can query the contents of the keystore
  - Example:
    - `gsk8capicmd_64 -cert -list -db ~/db2/db2keys.p12 -stashed`

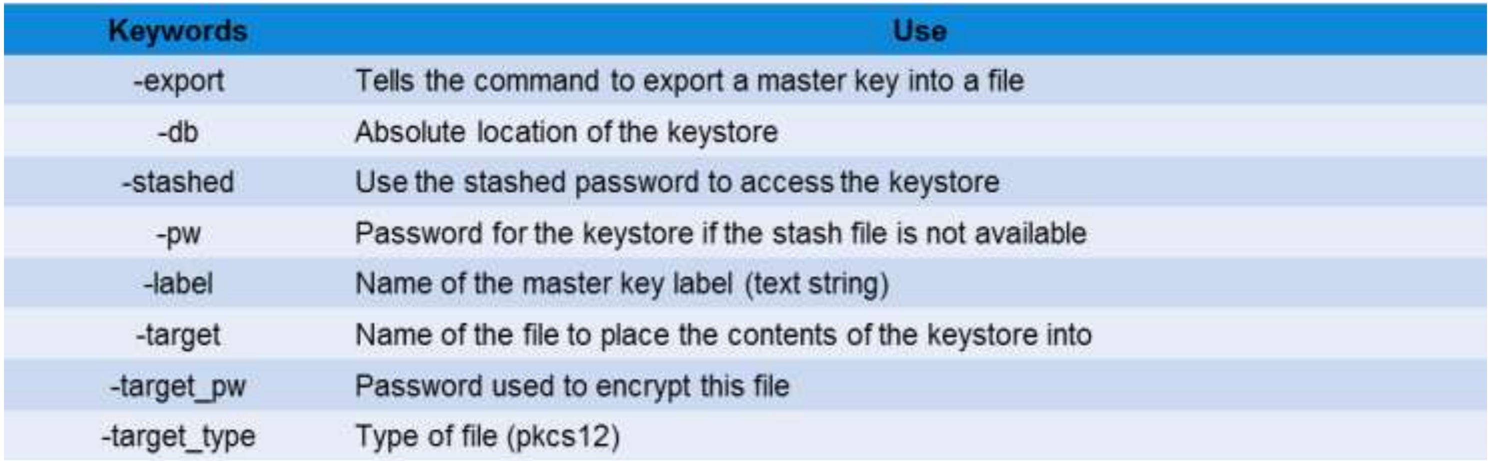


Common Keywords	Use
-db	Absolute location of the keystore
-stashed	Use the stashed password to access the keystore
-pw	Password for the keystore if the stash file is not available
Delete	**Use**
-delete -label	Name of the master key label (text string)
List	**Use**
-list	List all of the master keys in the keystore

Exporting Master Keys

- A secure method of transporting a key to another system to be imported into a keystore
  - Example:

    -
      ```
      gsk8capicmd_64 -cert -export -db ~/db2/db2keys.p12
          -stashed
         -label secret.key
         -target ~/db2/exportedkey.p12
         -target_type pkcs12
         -target_pw Str0ngPassw0rd
      ```

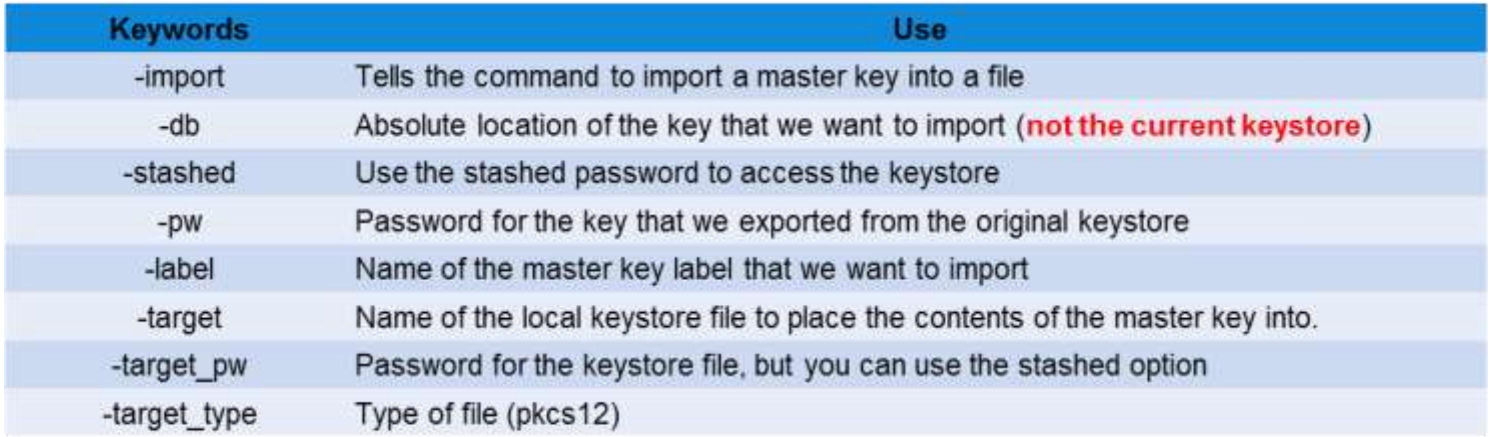


Keywords	Use
-export	Tells the command to export a master key into a file
-db	Absolute location of the keystore
-stashed	Use the stashed password to access the keystore
-pw	Password for the keystore if the stash file is not available
-label	Name of the master key label (text string)
-target	Name of the file to place the contents of the keystore into
-target_pw	Password used to encrypt this file
-target_type	Type of file (pkcs12)

Importing Master Keys

- A secure method of transporting a key to another system to be imported into a keystore (target is the destination keystore)
  - Example:

```
• gsk8capicmd_64 –cert –import -db ~/db2/exportedkey.p12
            -pw Str0ngPassw0rd
            -label secret.key
            –target ~/db2/db2keys.p12 –target_type pkcs12
            -stashed
```



Keywords	Use
-import	Tells the command to import a master key into a file
-db	Absolute location of the key that we want to import (**not the current keystore**)
-stashed	Use the stashed password to access the keystore
-pw	Password for the key that we exported from the original keystore
-label	Name of the master key label that we want to import
-target	Name of the local keystore file to place the contents of the master key into.
-target_pw	Password for the keystore file, but you can use the stashed option
-target_type	Type of file (pkcs12)

Registering the Keystore with DB2

- After creating a keystore file, the DB2 instance must be updated with the location and type of keystore
  - Two new configuration parameters
    - KEYSTORE_TYPE – Type of keystore being used (either NULL or PKCS12)
    - KEYSTORE_LOCATION – Absolute location of the keystore (or NULL if none)
- A DB2 instance can only have one keystore
  - The system could have keystores for other applications, but DB2 only supports one keystore at the instance level
- Best practice is to update both parameters simultaneously
  - Example:
    -
      ```
      UPDATE DBM CFG USING
             KEYSTORE_TYPE PKCS12
             KEYSTORE_LOCATION "/home/db2inst1/db2/db2keys.p12"
      ```
- To remove a keystore from an instance, set the values to NONE and NULL
  - Example:
    - `UPDATE DBM CFG USING KEYSTORE_TYPE NONE KEYSTORE_LOCATION NULL`

Agenda

Why Should We Encrypt Our Databases?

IBM DB2 Native Encryption Overview

Encryption key management

Encrypting DB2 databases

Backup and Restore

Utilities, diagnostics, and other considerations

Encrypting DB2 databases

- Once the keystore has been created and registered, and (optional) a MK created, you can encrypt a database
  - Example
    - `CREATE DATABASE mydb ENCRYPT`
    - `RESTORE DATABASE mydb from /home/db2inst1/db2 ENCRYPT`
- The default encryption algorithm is AES 256, but users can select other algorithms and key lengths if they so desire
  - Example

    -
      ```
      CREATE DATABASE mydb
          ENCRYPT CIPHER AES KEY LENGTH 128
      ```

    -
      ```
      CREATE DATABASE mydb
          ENCRYPT CIPHER 3DES KEY LENGTH 168
      ```

    -
      ```
      CREATE DATABASE mydb
          ENCRYPT CIPHER AES KEY LENGTH 256
          MASTER KEY LABEL mylabel
      ```

Encrypting DB2 databases

- The ENCRYPT keyword options on CREATE/RESTORE to new database command

```
--+----------------------------------------------------------------+
  '-ENCRYPT-+------------------------+----+----------------------+-'
            '-| Encryption Options |-'  '-| Master Key Options |-'

   Encryption Options
                        .-MODE--CBC-.
   |--CIPHER--+-AES--+--+-----------+--KEY LENGTH--key-length------|
              '-3DES-'

   Master Key Options
   |--MASTER KEY LABEL--label-name---------------------------------|
```

- KEY LENGTH
  - AES: 128, 192, or 256(default) bits
  - 3DES: 168 bits

Is my database encrypted ?

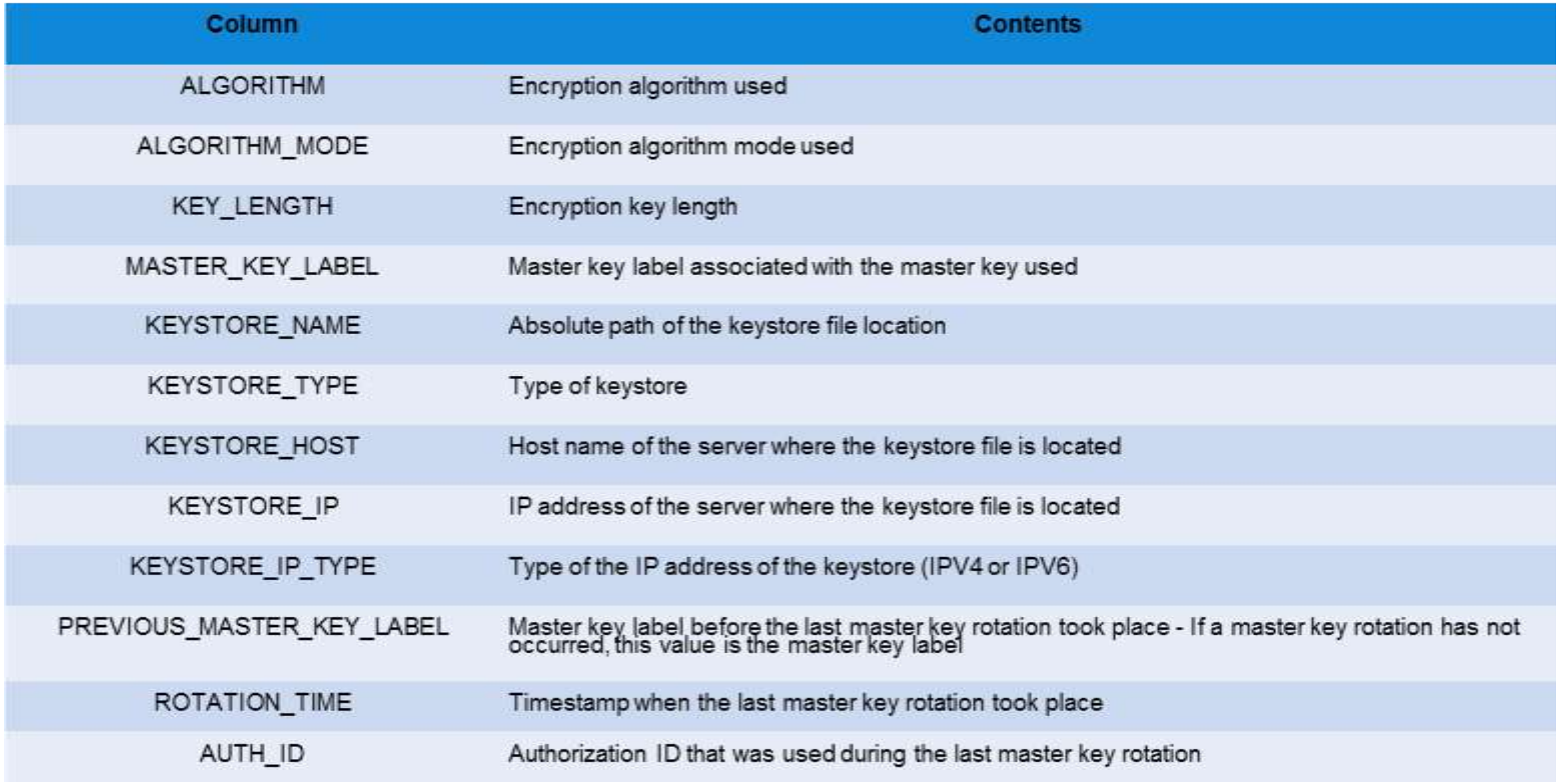
- To determine if a database is encrypted we can check the “Encrypted database” database configuration parameter
  - Example
    - `db2 get db cfg | grep -i encrypted`

```
Encrypted database = YES
```

I THINK WE SHOULD BUILD AN SQL DATABASE.
UH-OH
DOES HE UNDERSTAND WHAT HE SAID OR IS IT SOMETHING HE SAW IN A TRADE MAGAZINE AD?
WHAT COLOR DO YOU WANT THAT DATABASE?
I THINK MAUVE HAS THE MOST RAM.
S. Adams
E-mail: SCOTTADAMS@AOL.COM
11/17
© 1995 United Feature Syndicate, Inc.(NYC)

Current database encryption settings

- `SELECT * FROM TABLE(SYSPROC.ADMIN_GET_ENCRYPTION_INFO())`



Column	Contents
ALGORITHM	Encryption algorithm used
ALGORITHM_MODE	Encryption algorithm mode used
KEY_LENGTH	Encryption key length
MASTER_KEY_LABEL	Master key label associated with the master key used
KEYSTORE_NAME	Absolute path of the keystore file location
KEYSTORE_TYPE	Type of keystore
KEYSTORE_HOST	Host name of the server where the keystore file is located
KEYSTORE_IP	IP address of the server where the keystore file is located
KEYSTORE_IP_TYPE	Type of the IP address of the keystore (IPV4 or IPV6)
PREVIOUS_MASTER_KEY_LABEL	Master key label before the last master key rotation took place - If a master key rotation has not occurred, this value is the master key label
ROTATION_TIME	Timestamp when the last master key rotation took place
AUTH_ID	Authorization ID that was used during the last master key rotation

Master Key Rotation

- The process of changing encryption keys for compliance purposes
  - It requires decrypting any DEK encrypted with the old MK and then re-encrypting it with the new MK
  - **The data does not get re-encrypted!**

- The key rotation frequency depends on the compliance driver
  - This generally ranges from once every 3 months to once per year

- The key rotation requirement can be thought of as analogous to the requirement to change passwords every 90 days

Master Key Rotation

- The SYSPROC.ADMIN_ROTATE_MASTER_KEY procedure can be used to change the database key to comply with key rotation requirement
  - You must be connected to the database to run this command
    - `CALL SYSPROC.ADMIN_ROTATE_MASTER_KEY('newMasterKeyLabel')`

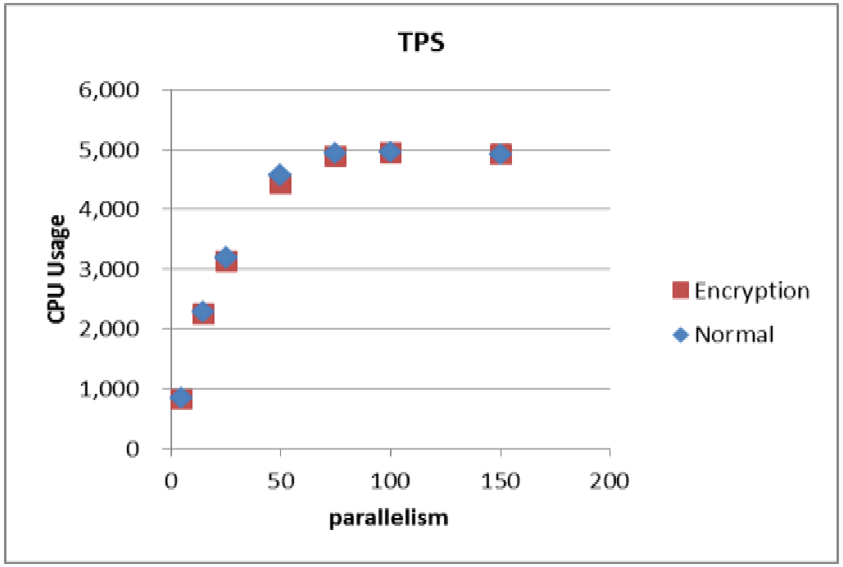
- The SYSPROC.ADMIN_ROTATE_MASTER_KEY procedure re-encrypts the DEK with the new MK
- DB2 will automatically generate the new MK unless you provide a MK label
- Key rotation is logged in the db2diag.log file:
    - `grep –A 3 "Key Rotation" ~/sqllib/db2dump/db2diag.log`

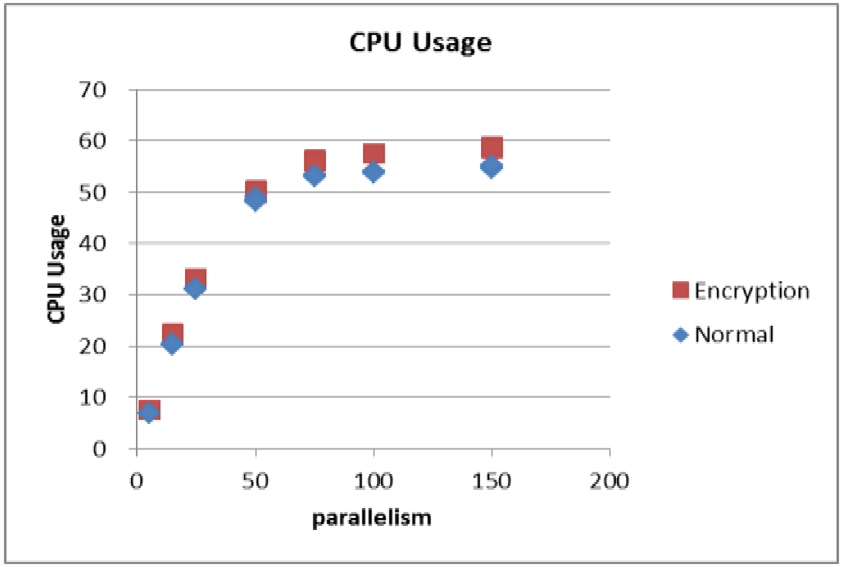
```
Key Rotation successful using label:
DATA #2 : String, 46 bytes
DB2_SYSGEN_db2inst1_SECRET_2015-02-09-05.03.12
```

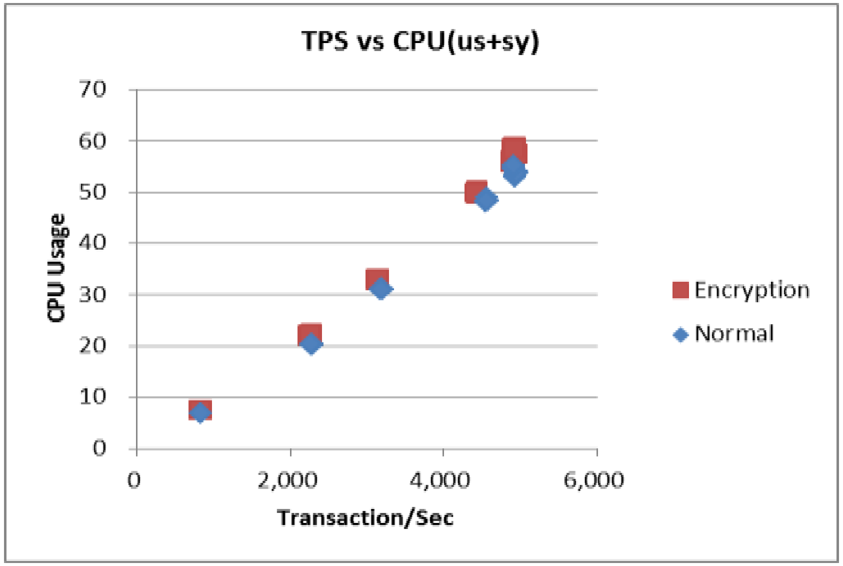
Data Encryption Key Rotation

- The reason an industry standard key encrypting key approach is to avoid a DEK rotation, however if there is a need to rotate the DEK:
  - Take an offline backup
  - Drop the database
  - Restore to a new encrypted database (this will generate a new DEK).

Performance overhead







TPS:
Almost NO difference b/w unencrypted DB vs. Encrypted DB

CPU Usage:
Unencrypted DB / Encrypted DB = 100-103%

TPS vs CPU(us+sy):
Encrypted DB needs a little more CPU(us) than Unencrypted DB per transaction.

DB2 internal benchmarks show that the encryption overhead is typically in the single digits for data warehouse workloads on systems with exploitable hardware acceleration for cryptographic operations. DB2 Native Encryption automatically detects and exploits a number of hardware acceleration for cryptographic operations built into modern CPUs such as Intel AES-NI on current Intel chips

Agenda

Why Should We Encrypt Our Databases?

IBM DB2 Native Encryption Overview

Encryption key management

Encrypting DB2 databases

Backup and Restore

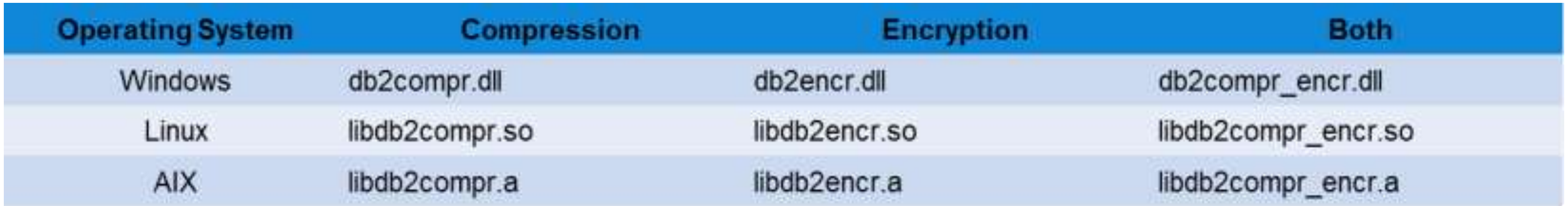
Utilities, diagnostics, and other considerations

Backup Encryption

- Example:

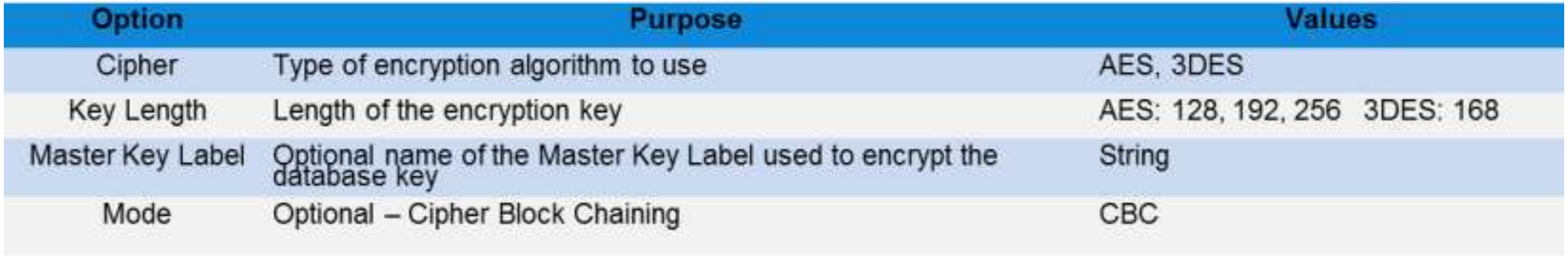
```
• BACKUP DATABASE mydb TO /HOME/DB2INST1/DB2
         ENCRYPT ENCRLIB 'libdb2encr.so'
         ENCROPTS 'Cipher=AES:Key Length=256‘
```

- ENCRLIB options



Operating System	Compression	Encryption	Both
Windows	db2compr.dll	db2encr.dll	db2compr_encr.dll
Linux	libdb2compr.so	libdb2encr.so	libdb2compr_encr.so
AIX	libdb2compr.a	libdb2encr.a	libdb2compr_encr.a

- ENCROPTS options: same as for CREATE/RESTORE to new database command



Option	Purpose	Values
Cipher	Type of encryption algorithm to use	AES, 3DES
Key Length	Length of the encryption key	AES: 128, 192, 256 3DES: 168
Master Key Label	Optional name of the Master Key Label used to encrypt the database key	String
Mode	Optional – Cipher Block Chaining	CBC

Enforce Automatic Backup Encryption

- ENCRLIB and ENCROPTS database configuration parameters
  - Set automatically for encrypted databases

  -
    ```
    $ db2 get db cfg | grep -i encryption
        Encryption Library for Backup (ENCRLIB) = libdb2encr.so
        Encryption Options for Backup(ENCROPTS) =
                         CIPHER=AES:MODE=CBC:KEY LENGTH=256
    ```
- Only SECADM can change/turn off the ENCRLIB, ENCROPTS db cfg parameters
- Only when ENCRLIB=NULL, ENCROPTS=NULL and no ENCRLIB and ENCROPS specified on the BACKUP DATABASE command a DBA can take a cleartext (not encrypted) backup.

Restore Encrypted Backup to Existing Database

- Restoring a backup by replacing an existing database requires no special parameters
  - Keystore must contain the MK that was used to encrypt this backup image
  - Cleartext databases with an encrypted backup restore to cleartext databases
    - RESTORE DATABASE mydb FROM /home/db2inst1/db2

- RESTORE will use the existing database encryption settings to **encrypt** the data being restored

- The encryption settings **can not** be changed when restoring into an existing database

Restore Encrypted Backup to a New Encrypted Database

- Restoring a backup to a new encrypted database requires that the **ENCRYPT** parameter be added to the command
  - DB2 needs to create the database before restoring the encrypted copy, and without the ENCRYPT keyword, the database would not be secure
  - Parameters for the ENCRYPT keyword are identical to creating an encrypted database

    -
      ```
      RESTORE DATABASE mydb FROM /home/db2inst1/db2
        ENCRLIB ‘libdb2encr.so’ ENCROPTS ‘Master Key
      Label=secret1.key’
        ENCRYPT
           CIPHER AES
           KEY LENGTH 128
           MASTER KEY LABEL secret2.key
      ```

Encrypted Backup Settings

- The RESTORE command can extract the backup encryption settings
  - The RESTORE command with the show master key details option will prompt the user if they want to overwrite an existing copy of the database
  - Accepting the overwrite will NOT overwrite the database

    -
      ```
      RESTORE DATABASE mydb FROM /home/db2inst1/db2
          ENCROPTS 'show master key details'
      ```
- Encryption settings from the backup will be placed into the db2dump directory
  - File with the following name will be generated
    `<DATABASE>.#.<instance>.<partition>.<timestamp>.masterkeydetails`
  - The encryption setting parameters are the same as for the database encryption
    - Algorithm
    - Key length
    - etc

Backup on primary site and Restore on backup site

- Create the database and do a backup

```
• CREATE DATABASE mydb ENCRYPT
• BACKUP DATABASE mydb TO /primary
```

- Extract the Master Key Label for the keystore

```
• gsk8capicmd -cert -export -db ~/db2/primary.p12 -stashed
             -label secret.key -target secret.p12
             -target_type pkcs12 -target_pw Str0ngPassw0rd
```

- Copy the master key to the backup site and add the key to the backup site keystore

```
• gsk8capicmd -cert -import -db secret.p12 -pw Str0ngPassw0rd
     -stashed -label secret.key -target ~/db2/backup.p12
                -target_type pkcs12
```

- Restore the database

```
• RESTORE DATABASE mydb FROM /backup ENCRYPT MASTER KEY LABEL
  secret.key
```

Agenda

Why Should We Encrypt Our Databases?

IBM DB2 Native Encryption Overview

Encryption key management

Encrypting DB2 databases

Backup and Restore

Utilities, diagnostics, and other considerations

HADR Considerations

- Normally both primary and secondary databases are encrypted
  - Possible to only have the primary or secondary encrypted
  - On HADR startup, an admin warning message will be produced

- Secondary site will be set up as new a database
  - Specify encryption options as part of the RESTORE command
  - Keystore needs to be available locally

YOU HAVEN'T HEARD WHAT THE PROBLEM IS YET; HOW CAN YOU RECOMMEND BUILDING A DATABASE TO SOLVE IT??
S. Adams E-mail: SCOTTADAMS@AOL.COM
WE ALWAYS BUILD A DATABASE.
AND WE'LL NEED COFFEE MUGS FOR THE PROJECT TEAM.
2/28/96 © 1996 United Feature Syndicate, Inc.(NYC)
THE PROBLEM IS THAT WE HAVE POOR PROCESSES.
THAT COULD BE THE SLOGAN ON OUR MUGS!

Tooling Changes

- Tools with encryption support
  - db2cklog
  - db2flsn
  - db2LogsForRfwd
  - db2ckbkp
  - db2adutl
  - db2dart

- These tools will use the keystore specified in the DBM CFG KEYSTORE_LOCATION parameter
  - Additional arguments used to connect to the keystore if the password is not stashed

```
-kspassword password
-kspassarg  fd:file_descriptor
            filename:file_name

 -ksprompt
```

Summary: Enabling Native Encryption on a New Database
Four steps

DB2 Native Encryption setup steps	Commands

Set the paths for the Global Security Kit (GSKit)	`export LD_LIBRARY_PATH=...`
	`export PATH=...`

Create a Keystore (using the GSKit command utility)	`gsk8capicmd_64 -keydb -create -db [keystore]...`

Configure DB2 instance with the Keystore information	`update dbm cfg using`
	`  keystore_type [keytype]`
	`  keystore_location [keypath]...`

Create the DB2 database using encryption	`create db [dbname] encrypt...`

Summary: Enabling Native Encryption on an Existing Database
Six steps

DB2 Native Encryption setup steps	Commands

Set the paths for the Global Security Kit (GSKit)	`export LD_LIBRARY_PATH=...`
	`export PATH=...`

Create a Keystore (using the GSKit command utility)	`gsk8capicmd_64 -keydb -create -db [keystore]...`

Configure DB2 instance with the Keystore information	`update dbm cfg using`
	`keystore_type [keytype]`
	`keystore_location [keypath]...`

Back up the database	`backup db [dbname]...`
Drop existing database	`drop db [dbname]`
Restore* the database with encryption	`restore db [dbname] encrypt ...`

* You can have on-line off-line backups and use RESTORE, RESTORE with ROLLFORWARD or RECOVER